AMENDMENTS TO THE SPECIFICATION

Please replace paragraph **[0005]** with the following amended paragraph:

[0005] It is increasingly common for processors to support instructions of variable instruction size. That is, some processors support both 16 and 32 bit instructions, and others also support 64 bit instructions. In this sense, an instruction *word* could be considered 16 bits, and instructions that are of bigger size could be considered to be multiple instruction words in length. The expression "instruction word" is a way to describe a smallest unit of usable instruction data, whether it be one or multiple bytes in size. System 100 includes loop buffers LB0 120 and LB1 121. In one embodiment system 100 is part of a processor that supports variable instruction sizes. Thus, note that loop buffers will be inefficient in their use of resources. The bus to loop buffers LB0 and LB1 are of size *y*, which may be four times the width of some instructions (64 = 16 * 4). Thus, even when a 2 byte instruction is to be input into LB0 **[[~~130~~]] 120**, an 8-byte storage location is dedicated to holding the instruction. Because instruction latch 130 may receive instructions from loop buffers LB0 120 and LB1 121 as an alternative to receiving instructions from alignment buffer 110, the instruction data received from LB0 120 and LB1 121 should also be aligned as it would be from alignment buffer 110. The need to align the instruction data for transmission to instruction latch 130 creates the wasteful use of resources, where parts of the loop buffers may be left empty to ensure proper alignment of instructions.

Please replace paragraph **[0024]** with the following amended paragraph:

[0024] Instructions are read from L0 cache 320 to instruction latch 330 and passed on to decoder **[[~~330~~]] 340**, similar to that discussed above with reference to these elements. The alignment of contiguously-stored variable-sized instructions is discussed below for embodiments of a system having L0 cache **[[~~230~~]] 320** that supports variable instruction sizes.

Please replace paragraph **[0053]** with the following amended paragraph:

[0053] L0 cache **[[~~721~~]]** **701** will continue to read instructions in this fashion. Note that when PC is set to, for example, instr7, the current instruction will be an instruction that straddles the storage cells. This is because instruction instr7 includes two words, instr7a and instr7b, with one at one line of its storage cell, and the other at a different line of its storage cell. A read beginning at memory location 727 will result in permutation logic 740 receiving, in order, 728 (instr7b), 729 (instr8a), 730 (instr8b), 727 (instr7a). Permutation logic 740 will permute this group of instruction data words to align the instruction, the PC will be updated to read next at memory location 729 (instr8).

Please replace paragraph **[0054]** with the following amended paragraph:

[0054] Now consider a case in which the instruction data stored in memory locations 721 (instr2) to 728 (instr7) constitutes the instructions of a software loop. A loop top pointer and a loop bottom pointer in L0 control 760 may be used to define the loop window. Because the loop window is within the size of L0 cache **[[~~721~~]]** **701**, the loop can be executed directly out of L0 cache **[[~~721~~]]** **701** without requiring a read from the L1 memory system. Execution of the loop proceeds with instructions executed from 721/722, 723, 724, 725, 726, 727/728, in that order. At instruction instr7, the PC is reset to memory location 721 and execution is continued without incurring a branch delay because the instruction at the top of the loop are found in L0 cache 701, and without incurring penalties for reading from L1 because all loop instructions are found in L0 cache 701.